RECORDING MEDIUM
100
READOUT UNIT
28
DEMODULATION
29
ECC DECODER
30
AV STREAM
SOURCE DEPACKETIZER
31
DEMULTIPLEXER
26
AV DECODER
27
VIDEO OUTPUT
32
AUDIO OUTPUT
33
3
V
A
S
DATA BASE
CONTROLLER
23
USER INTERFACE INPUT/ OUTPUT
24
DATA BASE
MUX STREAM ANALYSIS
18
SOURCE PACKETIZER
19
AV STREAM
ECC ENCODER
20
WRITE UNIT
22
MODULATION
21
2
ANALYSIS
14
AV ENCODER
15
MULTIPLEXER
16
17
TRANSPORT STREAM
VIDEO INPUT
11
AUDIO INPUT
12
DIGITAL INTERFACE INPUT
DIGITAL TV TUNER INPUT
13

FIG.1

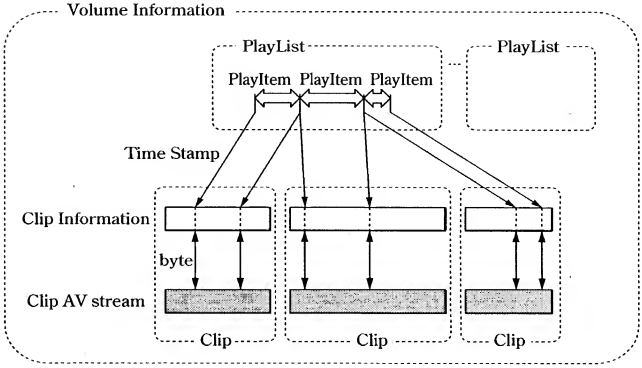

Volume Information
PlayList
PlayList
PlayItem PlayItem PlayItem
Time Stamp
Clip Information
byte
Clip AV stream
Clip
Clip
Clip

FIG.2

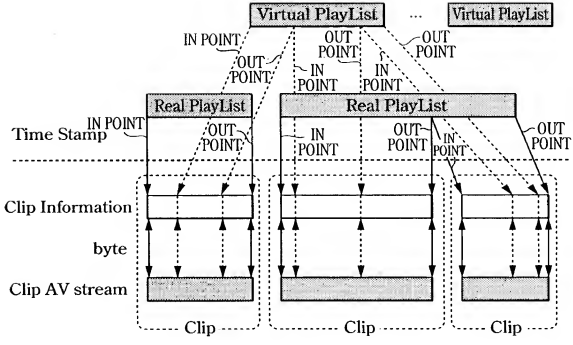

Virtual PlayList
...
Virtual PlayList
IN POINT
OUT POINT
OUT POINT
OUT POINT
IN POINT
IN POINT
Real PlayList
Real PlayList
IN POINT
Time Stamp
OUT POINT
IN POINT
OUT POINT
IN POINT
OUT POINT
Clip Information
byte
Clip AV stream
Clip
Clip
Clip

FIG.3

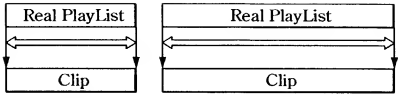


FIG.4A

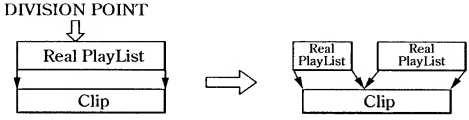


FIG.4B

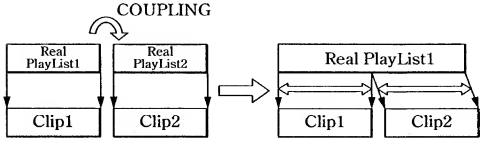


FIG.4C

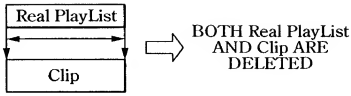


FIG.5A

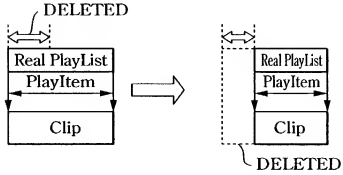


FIG.5B

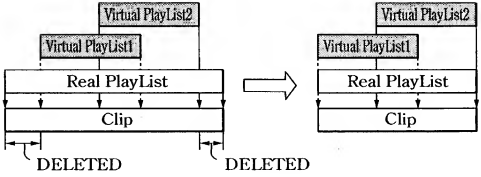


FIG.5C

FIG.6A

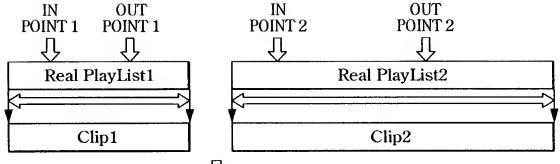
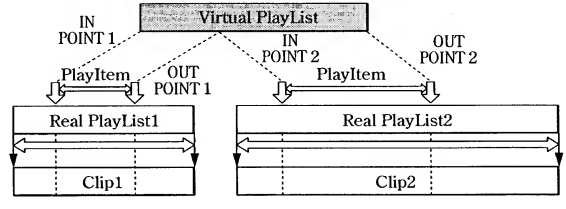


FIG.6B



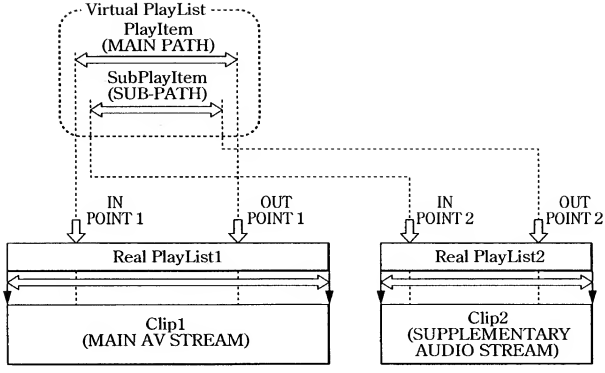

Virtual PlayList
PlayItem
(MAIN PATH)
SubPlayItem
(SUB-PATH)
IN
POINT 1
OUT
POINT 1
IN
POINT 2
OUT
POINT 2
Real PlayList1
Real PlayList2
Clip1
(MAIN AV STREAM)
Clip2
(SUPPLEMENTARY
AUDIO STREAM)

FIG.7

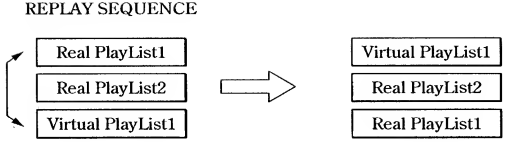

REPLAY SEQUENCE
Real PlayList1
Real PlayList2
Virtual PlayList1
Virtual PlayList1
Real PlayList2
Real PlayList1

FIG.8

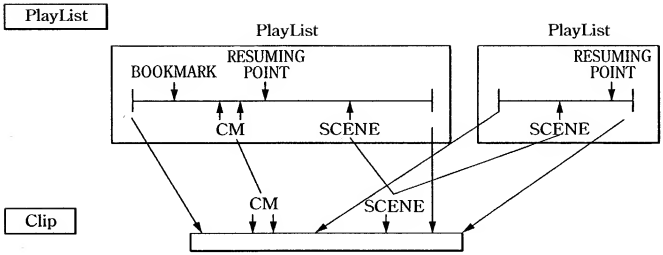

PlayList
PlayList
PlayList
PlayList
BOOKMARK
RESUMING POINT
RESUMING POINT
CM
SCENE
SCENE
CM
SCENE
Clip

FIG.9

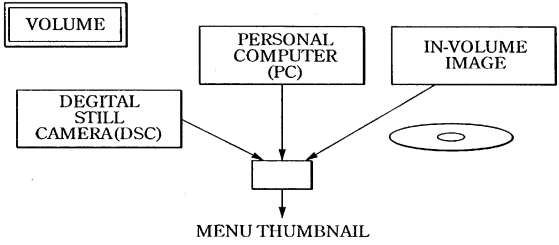

VOLUME
PERSONAL COMPUTER (PC)
IN-VOLUME IMAGE
DEGITAL STILL CAMERA(DSC)
MENU THUMBNAIL

FIG.10

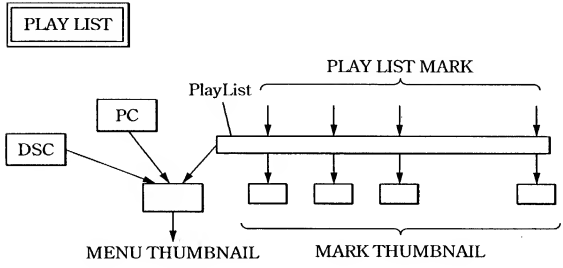

PLAY LIST
PLAY LIST MARK
PlayList
PC
DSC
MENU THUMBNAIL
MARK THUMBNAIL

FIG.11

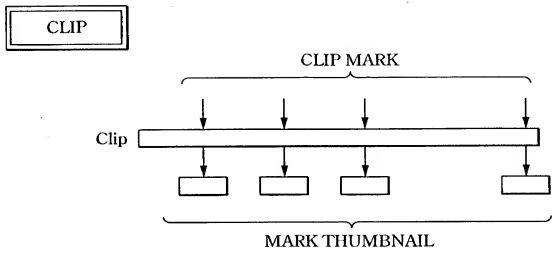

CLIP
CLIP MARK
Clip
MARK THUMBNAIL

FIG.12

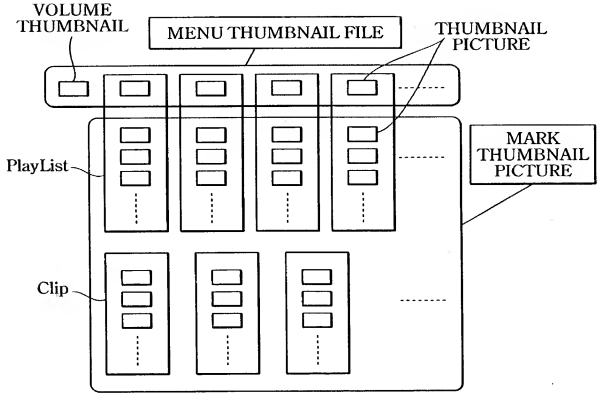

VOLUME THUMBNAIL
MENU THUMBNAIL FILE
THUMBNAIL PICTURE
MARK THUMBNAIL PICTURE
PlayList
Clip

FIG.13

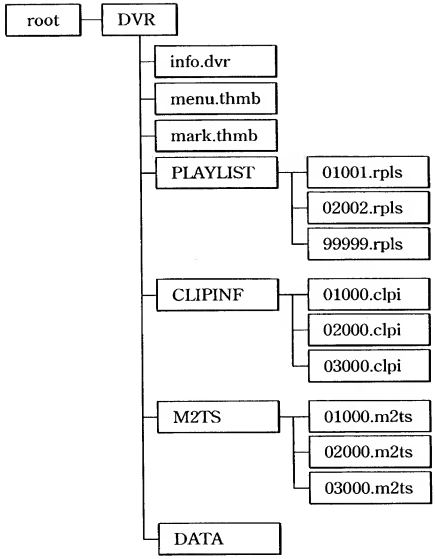

root
DVR
info.dvr
menu.thmb
mark.thmb
PLAYLIST
01001.rpls
02002.rpls
99999.rpls
CLIPINF
01000.clpi
02000.clpi
03000.clpi
M2TS
01000.m2ts
02000.m2ts
03000.m2ts
DATA

FIG.14

SYNTAX	NUMBER OF BYTES	ABBREVIATION
info.dvr {		
TableOfPlayLists_Start_address	32	uimsbf
MakersPrivateData_Start_address	32	uimsbf
reserved	192	bslbf
DVRVolume()		
for (i=0;i<N1;i++){		
padding_word	16	bslbf
}		
TableOfPlayLists()		
for (i=0;i<N2;i++){		
padding_word	16	bslbf
}		
MakersPrivateData()		
}		

FIG.15

SYNTAX	NUMBER OF BYTES	ABBREVIATION
DVRVolume(){		
version_number	8*4	bslbf
length	32	uimsbf
ResumeVolume()		
UIAppInfoVolume()		
}		

FIG.16

SYNTAX	NUMBER OF BYTES	ABBREVIATION
ResumeVolume(){		
reserved	15	bslbf
valid_flag	1	bslbf
resume_PlayList_name	8*10	bslbf
}		

FIG.17

SYNTAX	NUMBER OF BYTES	ABBREVIATION
UIAppInfoVolume(){		
character_set	8	bslbf
name_length	8	uimsbf
Volume_name	8*256	bslbf
reserved	15	bslbf
Volume_protect_flag	1	bslbf
PIN	8*4	bslbf
ref_thumbnail_index	16	uimsbf
reserved_for_future_use	256	bslbf
}		

FIG.18

VALUE	CHARACTER LETTER ENCODING
0x00	Reserved
0x01	ISO/IEC 646 (ASCII)
0x02	ISO/IEC 10646-1 (Unicode)
0x03-0xff	Reserved

FIG.19

SYNTAX	NUMBER OF BYTES	ABBREVIATION
TableOfPlayLists(){		
version_number	8*4	bslbf
length	32	uimsbf
number_of_PlayLists	16	uimsbf
for (i=0; i<*number_of_PlayLists*; i++){		
PlayList_file_name	8*10	bslbf
}		
}		

FIG.20

SYNTAX	NUMBER OF BYTES	ABBREVIATION
TableOfPlayLists(){		
version_number	8*4	bslbf
length	32	uimsbf
number_of_PlayLists	16	uimsbf
for (i=0; i<*number_of_PlayLists*; i++){		
PlayList_file_name	8*10	bslbf
UIAppInfoPlayList()		
}		
}		

FIG.21

SYNTAX	NUMBER OF BYTES	ABBREVIATION
MakersPrivateData(){		
version_number	8*4	bslbf
length	32	uimsbf
if (length !=0){		
mpd_blocks_start_address	32	uimsbf
number_of_maker_entries	16	uimsbf
mpd_block_size	16	uimsbf
number_of_mpd_blocks	16	uimsbf
reserved	16	bslbf
for (i=0; i<*number_of_maker_entries*; i++){		
maker_ID	16	uimsbf
maker_model_code	16	uimsbf
start_mpd_block_number	16	uimsbf
reserved	16	bslbf
mpd_length	32	uimsbf
}		
stuffing_bytes	8*2*L1	bslbf
for(j=0; j<*number_of_mpd_blocks*; j++){		
mpd_block	mpd_block_ size*1024*8	
}		
}		
}		

FIG.22

SYNTAX	NUMBER OF BYTES	ABBREVIATION
xxxxx.rpls / yyyyy.vpls {		
PlayListMark_Start_address	32	uimsbf
MakersPrivateData_Start_address	32	uimsbf
reserved	192	bslbf
PlayList()		
for (i=0;i<N1;i++){		
padding_word	16	bslbf
}		
PlayListMark()		
for (i=0;i<N2;i++){		
padding_word	16	bslbf
}		
MakersPrivateData()		
}		

FIG.23

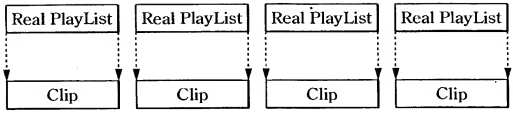


FIG.24A

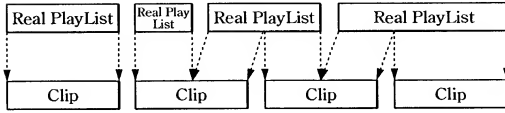


FIG.24B

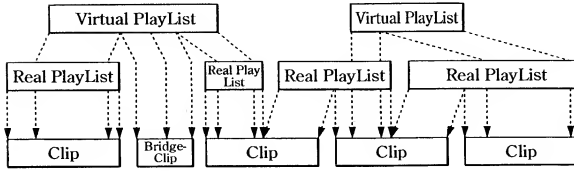


FIG.24C

SYNTAX	NUMBER OF BYTES	ABBREVIATION
PlayList(){		
version_number	8*4	bslbf
length	32	uimsbf
PlayList_type	8	uimsbf
CPI_type	1	bslbf
reserved	7	bslbf
UIAppInfoPlayList()		
number_of_PlayItems // main path	16	uimsbf
if (<*Vertual PlayList*>){		
number_of_SubPlayItems // sub path	16	uimsbf
}else{		
reserved	16	bslbf
}		
for (*PlayItem_id=0;* *PlayItem_id<nymber_of_PlayItems;* *PlayItem_id++*){		
PlayItem() //main path		
}		
if (<*Virtual PlayList*>){		
if (CPI_type==0 && PlayList_type==0){		
for (i=0; i<*number_of _SubPlayItems*; i++)		
SubPlayItem() //sub path		
}		
}		
}		

FIG.25

PlayList_type	MEANING
0	PLAY LIST FOR AV RECORDING ALL CLIPS REFERENCED IN THIS PLAY LIST MUST CONTAIN ONE OR MORE VIDEO STREAMS
1	PLAY LIST FOR AUDIO RECORDING ALL CLIPS REFERENCED IN THIS PLAYLIST MUST CONTAIN ONE OR MORE AUDIO STREAMS AND MUST NOT CONTAIN VIDEO STREAMS
2-255	reserved

FIG.26

SYNTAX	NUMBER OF BYTES	ABBREVIATION
UIAppInfoPlayList2(){		
character_set	8	bslbf
name_length	8	uimsbf
PlayList_name	8*256	bslbf
reserved	8	bslbf
record_time_and_date	4*14	bslbf
reserved	8	bslbf
duration	4*6	bslbf
valid_period	4*8	bslbf
maker_id	16	uimsbf
maker_code	16	uimsbf
reserved	11	bslbf
playback_control_flag	1	bslbf
write_protect_flag	1	bslbf
is_played_flag	1	bslbf
archive	2	bslbf
ref_thumbnail_index	16	uimsbf
reserved_for_future_use	256	bslbf
}		

FIG.27

write_protect_flag	MEANING
0b	THE PlayList CAN BE ERASED FREELY
1b	THE PlayList CONTENTS SHOULD NOT BE ERASED NOR CHANGED EXCEPT write-protect-flag

FIG.28A

is_played_flag	MEANING
0b	THE PlayList HAS NOT BEEN REPRODUCED SINCE ITS RECORDING
1b	THE PlayList WAS ONCE REPRODUCED SINCE ITS RECORDING

FIG.28B

archive	MEANING
00b	NO MEANING DEFINED
01b	ORIGINAL
10b	COPY
11b	reserved

FIG.28C

GLOBAL TIME AXIS OF PlayList
PlayList (EP_map type)
PREVIOUS PlayItem
CURRENT PlayItem
connection _condition
IN_time(PTS) STC_sequence_id
OUT_time(PTS) STC_sequence_id
Clip1
STC_sequence
IN_time(PTS) STC_sequence_id
OUT_time(PTS) STC_sequence_id
Clip2
STC sequence

FIG.29

GLOBAL TIME
AXIS OF PlayList
Virtual
PlayList
(EP_map
type)
PREVIOUS
PlayItem
CURRENT
PlayItem
IN_time1(PTS)
STC_sequence_id
OUT_time1(PTS)
IN_time2(PTS)
OUT_time2(PTS)
STC_sequence_id
Clip1
STC_sequence
Bridge-Clip
STC_sequence
(STC_sequence_id=0)
STC_sequence
(STC_sequence_id=1)
Clip2
STC_sequence

FIG.30

GLOBAL TIME
AXIS OF PlayList
PlayList
(T_map
type)
PREVIOUS
PlayItem
CURRENT
PlayItem
connection
condition
shall be 0
IN_time
(Arrival Time)
OUT_time
(Arrival Time)
Clip1
TU map time axis
IN_time
(Arrival Time)
OUT_time
(Arrival Time)
Clip2
TU map time axis

FIG.31

SYNTAX	NUMBER OF BYTES	ABBREVIATION
PlayItem(){		
Clip_information_file_name	8*10	bslbf
reserved	24	bslbf
STC_sequence_id	8	uimsbf
IN_time	32	uimsbf
OUT_time	32	uimsbf
reserved	14	bslbf
connection_condition	2	bslbf
if (<Virtual PlayList>){		
if (*connection_condition*=='*10*'){		
BridgeSequenceInfo()		
}		
}		
}		

FIG.32

CPI_type in the PlayList()	SEMANTICS OF IN_time
EP_map type	IN_time MUST INDICATE UPPER 32 BITS OF 33 BIT LENGTH CORRESPONDING TO FIRST PRESENTATION UNIT IN PlayItem
TU_map type	IN_time MUST BE TIME ON TU_map_time_axis, AND MUST BE ROUNDED TO time_unit PRECISION. IN-time IS CALCULATED BY FOLLOWING EQUATION: $IN\_time = TU\_start\_time \%2^{32}$

FIG.33

CPI_type in the PlayList()	SEMANTICS OF OUT_time
EP_map type	OUT_time MUST INDICATE UPPER 32 BITS OF THE VALUE OF Presentation_end_TS CALCULATED BY FOLLOWING EQUTION: Presentation_end_TS = PTS_out+AU_duration WHERE PTS_out IS 33-BIT LONG PTS CORRESPONDING TO LAST PRESENTATION UNIT IN PlayItem. AU_duration IS 90 kHz-DISPLAY TIME OF LAST PRESENTATION UNIT.
TU_map type	OUT_time MUST BE TIME ON TU_map_time_axis AND BE ROUNDED TO time_unit PRECISION. OUT_time IS CALCULATED BY FOLLOWING EQUATION: $OUT\_time = TU\_start\_time \%2^{32}$

FIG.34

connection _condition	MEANING
00	· CONNECTION OF PREVIOUS PlayItem TO CURRENT PlayItem IS NOT SURE AS TO SEAMLESS REPLAY. · IF CPI_type OF PlayList IS TU_map type, THIS VALUE MUST BE SET IN connection_condition.
01	· THIS STATE IS ALLOWED ONLY WHEN CPI_type OF PlayList IS EP_map type. · PREVIOUS PlayItem AND CURRENT PlayItem INDICATE DIVISION BECAUSE OF NON-CONTINUOUS POINT OF SYSTEM TIMEBASE (STC BASE).
10	· THIS STATE IS ALLOWED ONLY WHEN CPI_type OF PlayList IS EP_map type. · THIS STATE IS ALLOWED ONLY FOR Virtual PlayList. · CONNECTION OF PREVIOUS PlayItem TO CURRENT PlayItem IS SURE AS TO SEAMLESS REPLAY. · PREVIOUS PlayItem IS CONNECTED TO CURRENT PlayItem USING BridgeSequence. DVR MPEG-2 TRANSPORT STREAM MUST OBEY DVR-STD AS LATER DESCRIBED.
11	· THIS STATE IS ALLOWED ONLY WHEN CPI_type OF PlayList IS EP_map type. · CONNECTION OF PREVIOUS PlayItem TO CURRENT Play Item IS SURE AS TO SEAMLESS REPLAY. · PREVIOUS PlayItem IS CONNECTED TO CURRENT PlayItem WITHOUT USING BridgeSequence. DVR MPEG-2 TRANSPORT STREAM MUST OBEY DVR-STD AS LATER DESCRIBED.

FIG.35

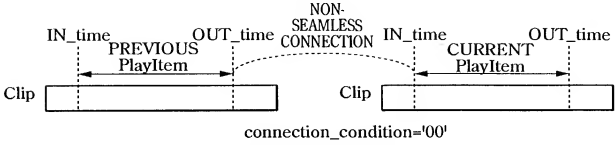


FIG.36A

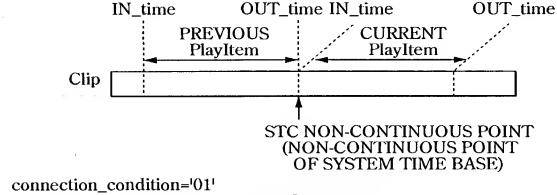


FIG.36B

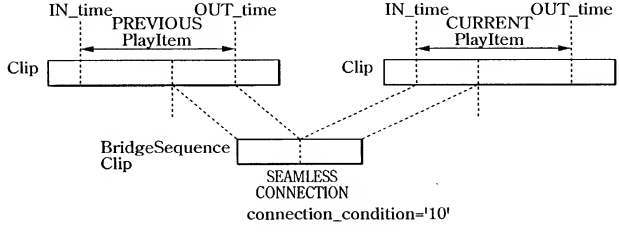


FIG.36C

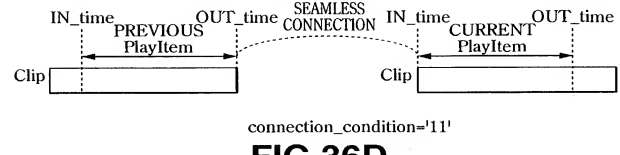


FIG.36D

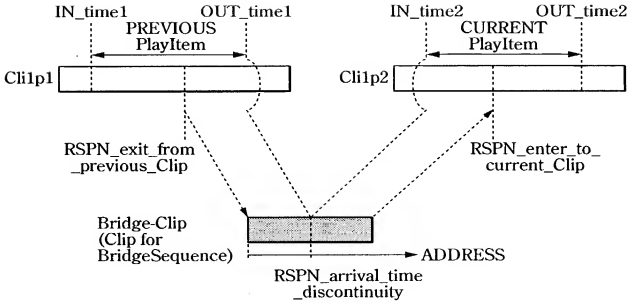

IN_time1
OUT_time1
PREVIOUS
PlayItem
IN_time2
OUT_time2
CURRENT
PlayItem
Cli1p1
Cli1p2
RSPN_exit_from
_previous_Clip
RSPN_enter_to_
current_Clip
Bridge-Clip
(Clip for
BridgeSequence)
ADDRESS
RSPN_arrival_time
_discontinuity

FIG.37

SYNTAX	NUMBER OF BYTES	ABBREVIATION
BridgeSequenceInfo() {		
Bridge_Clip_information_file_name	8*10	bslbf
RSPN_exit_from_previous_Clip	32	uimsbf
RSPN_enter_to_current_Clip	32	uimsbf
}		

FIG.38

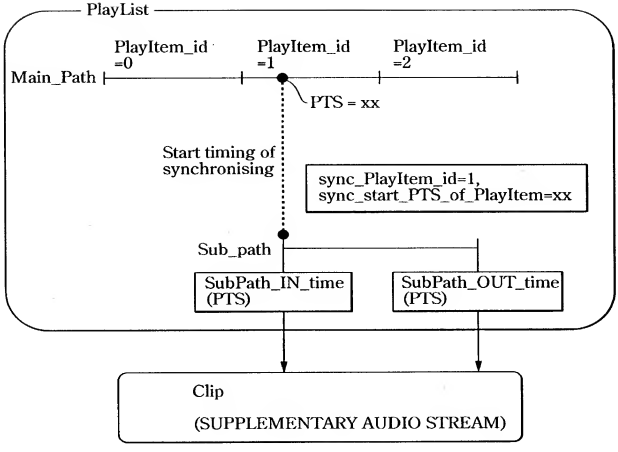

PlayList
PlayItem_id =0
PlayItem_id =1
PlayItem_id =2
Main_Path
PTS = xx
Start timing of synchronising
sync_PlayItem_id=1, sync_start_PTS_of_PlayItem=xx
Sub_path
SubPath_IN_time (PTS)
SubPath_OUT_time (PTS)
Clip
(SUPPLEMENTARY AUDIO STREAM)

FIG.39

SYNTAX	NUMBER OF BYTES	ABBREVIATION
SubPlayItem(){		
Clip_Information_file_name	8*10	bslbf
SubPath_type	8	bslbf
sync_PlayItem_id	8	uimsbf
sync_start_PTS_of_PlayItem	32	uimsbf
SubPath_IN_time	32	uimsbf
SubPath_OUT_time	32	uimsbf
}		

FIG.40

SubPath_type	MEANING
0x00	Auxiliary audio steam path
0x01-0xff	reserved

FIG.41

SYNTAX	NUMBER OF BYTES	ABBREVIATION
PlayListMark(){		
version_number	8*4	bslbf
length	32	uimsbf
number_of_PlayList_marks	16	uimsbf
for (i=0;i<*number_of_PlayList_marks*;i++){		
reserved	8	bslbf
mark_type	8	bslbf
mark_time_stamp	32	uimsbf
PlayItem_id	8	uimsbf
reserved	24	uimsbf
character_set	8	bslbf
name_length	8	uimsbf
mark_name	8*256	bslbf
ref_thumbnail_index	16	uimsbf
}		
}		

FIG.42

Mark_type	MEANING	COMMENT
0x00	resume-mark	REPLAY RESUME POINT. THE NUMBER OF REPLAY RESURE POINTS DEFINED IN PlayListMark() MUST BE 0 OR 1.
0x01	book-mark	REPLAY ENTRY POINT OF PlayList. THIS MARK CAN BE SET BY USER AND USED AS MARK SPECIFYING START POINT OF FAVORITE SCENE.
0x02	skip-mark	SKIP MARK POINT. PLAYER SKIPS PROGRAM FROM THIS POINT TO THE END OF PROGRAM. THE NUMBER OF SKIP MARK POINTS DEFINED IN PlayListMark() MUST BE 0 RO 1.
0x03-0x8F	reserved	
0x90-0xFF	reserved	Reserved for ClipMark()

FIG.43

CPI_type in the PlayList()	SEMANTICS OF mark_time_stamp
EP_map type	mark_time_stamp MUST INDICATE UPPER 32 BITS OF 33 BIT LENGTH PTS CORRESPONDING TO PRESENTATION UNIT REFERENCED BY MARK.
TU_map type	mark_time_stamp MUST BE TIME ON TU_map_time_axis AND MUST BE ROUNDED TO time_unit PRECISION. mark_time_stamp IS CALCULATED BY FOLLOWING EQUATION: $mark\_time\_stamp = TU\_start\_time \%2^{32}$

FIG.44

SYNTAX	NUMBER OF BYTES	ABBREVIATION
zzzzz.clpi {		
STC_Info_Start_address	32	uimsbf
ProgramInfo_Start_address	32	uimsbf
CPI_Start_address	32	uimsbf
ClipMark_Start_address	32	uimsbf
MakersPrivateData_Start_address	32	uimsbf
reserved	96	bslbf
ClipInfo()		
for (i=0;i<N1;i++){		
padding_word	16	bslbf
}		
STC_Info()		
for (i=0;i<N2;i++){		
padding_word	16	bslbf
}		
ProgramInfo()		
for (i=0;i<N3;i++){		
padding_word	16	bslbf
}		
CPI()		
for (i=0;i<N4;i++){		
padding_word	16	bslbf
}		
ClipMark()		
for (i=0;i<N5;i++){		
padding_word	16	bslbf
}		
MakersPrivateData()		
}		

FIG.45

SYNTAX	NUMBER OF BYTES	ABBREVIATION
ClipInfo(){		
version_number	8*4	bslbf
length	32	uimsbf
Clip_stream_type	8	bslbf
offset_SPN	32	uimsbf
TS_recording_rate	24	uimsbf
reserved	8	bslbf
record_time_and_date	4*14	bslbf
reserved	8	bslbf
duration	4*6	bslbf
reserved	7	bslbf
time_controlled_flag	1	bslbf
TS_average_rate	24	uimsbf
if (Clip_stream_type==1) // Bridge-Clip AV stream		
RSPN_arrival_time_discontinuity	32	uimsbf
else		
reserved	32	bslbf
reserved_for_system_use	144	bslbf
reserved	11	bslbf
is_format_identifier_valid	1	bslbf
is_original_network_ID_valid	1	bslbf
is_transport_stream_ID_valid	1	bslbf
is_service_ID_valid	1	bslbf
is_country_code_valid	1	bslbf
format_identifier	32	bslbf
original_network_ID	16	uimsbf
transport_stream_ID	16	uimsbf
service_ID	16	uimsbf
country_code	24	bslbf
stream_format_name	16*8	bslbf
reserved_for_fortune_use	256	bslbf
}		

FIG.46

Clip_stream_type	MEANING
0	Clip AV STREAM
1	Bridge-Clip AV STREAM
2-255	Reserved

FIG.47

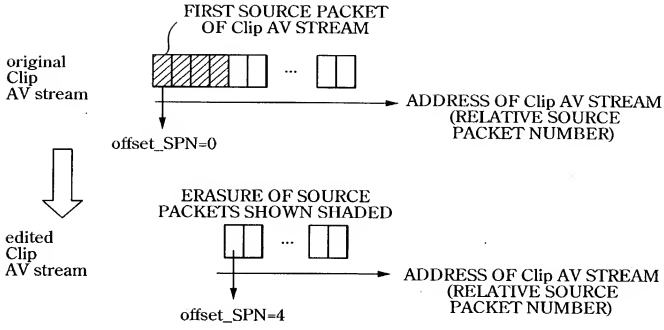

FIRST SOURCE PACKET
OF Clip AV STREAM
original
Clip
AV stream
ADDRESS OF Clip AV STREAM
(RELATIVE SOURCE
PACKET NUMBER)
offset_SPN=0
ERASURE OF SOURCE
PACKETS SHOWN SHADED
edited
Clip
AV stream
ADDRESS OF Clip AV STREAM
(RELATIVE SOURCE
PACKET NUMBER)
offset_SPN=4

FIG.48

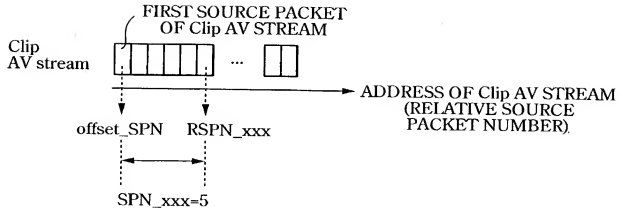


FIG.49

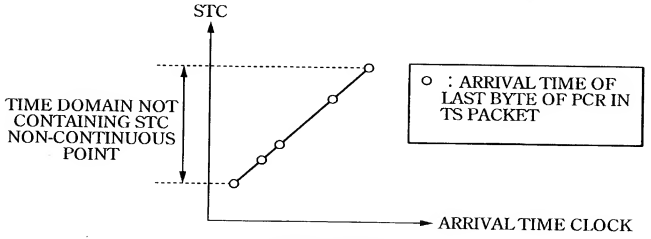


FIG.50A

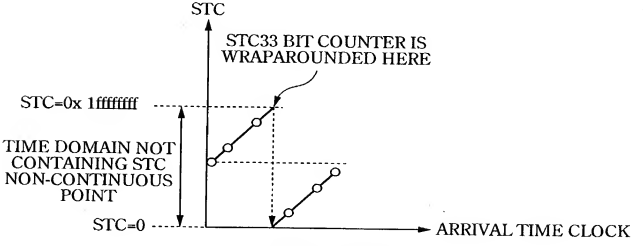


FIG.50B

FIRST SOURCE PACKET
OUTPUT SOURCE PACKET
Clip AV STREAM
STC NON-CONTINUOUS POINT
STC_sequence #0
STC_sequence #1
STC_ sequence #2
LAST SOURCE PACKET
ARRIVAL TIME CLOCK
RELATIVE SOURCE PACKET NUMBER
RSPN_ STC_start #0 =0
RSPN_ STC_start #1 =xx
RSPN_ STC_start #2 =yy
num_of_STC_sequences =3

FIG.51

SYNTAX	NUMBER OF BYTES	ABBREVIATION
STC_Info(){		
version_number	8*4	bslbf
length	32	uimsbf
if (length !=0){		
reserved	8	bslbf
num_of_STC_sequences	8	uimsbf
for (*STC_sequence_id=0;* *STC_sequence_id<num_of_STC_sequences;* *STC_sequence_id++*){		
resereved	32	bslbf
RSPN_STC_start	32	uimsbf
}		
}		
}		

FIG.52

FIG.53

SYNTAX	NUMBER OF BYTES	ABBREVIATION
ProgramInfo(){		
version_number	8*4	bslbf
length	32	uimsbf
if (length !=0){		
reserved	8	bslbf
number_of_program_sequences	8	uimsbf
for (i=0;i<*number_of_program_sequences*;i++){		
RSPN_program_sequence_start	32	uimsbf
reserved	48	bslbf
PCR_PID	16	bslbf
number_of_videos	8	uimsbf
number_of_audios	8	uimsbf
for (k=0;k<*number_of_videos*;k++){		
video_stream_PID	16	bslbf
VideoCodingInfo()		
}		
for (k=0;k<*number_of_audios*;k++){		
audio_stream_PID	16	bslbf
AudioCodingInfo()		
}		
}		
}		
}		

FIG.54

SYNTAX	NUMBER OF BYTES	ABBREVIATION
VideoCodingInfo() {		
video_format	8	uimsbf
frame_rate	8	uimsbf
display_aspect_ratio	8	uimsbf
reserved	8	bslbf
}		

FIG.55

video_format	MEANING
0	480i
1	576i
2	480p(including 640×480p format)
3	1080i
4	720p
5	1080p
6-254	reserved
255	No information

FIG.56

frame_rate	MEANING
0	forbidden
1	24 000/1001 (23.976...)
2	24
3	25
4	30 000/1001 (29.97..)
5	30
6	50
7	60 000/1001 (59.94..)
8	60
9-254	reserved
255	No information

FIG.57

display_aspect_ratio	MEANING
0	forbidden
1	reserved
2	4:3 display aspect ratio
3	16:9 display aspect ration
4-254	reserved
255	No information

FIG.58

SYNTAX	NUMBER OF BYTES	ABBREVIATION
AudioCodingInfo() {		
audio_format	8	uimsbf
audio_component_type	8	uimsbf
sampling_frequency	8	uimsbf
reserved	8	bslbf
}		

FIG.59

audio_coding	MEANING
0	MPEG-1 audio layer I or II
1	Dolby AC-3 audio
2	MPEG-2 AAC
3	MPEG-2 multi-channel audio, backward compatible to MPEG-1
4	SESF LPCM audio
5-254	reserved
255	No information

FIG.60

audio_component_type	MEANING
0	single mono channel
1	dual mono channel
2	stereo (2-channel)
3	multi-lingual, multi-channel
4	surround sound
5	audio description for the visually impaired
6	audio for the hard of hearing
7-254	reserved
255	No information

FIG.61

sampling_frequency	MEANING
0	48 kHz
1	44.1 kHz
2	32 kHz
3-254	reserved
255	No information

FIG.62

PlayList
(EP_map type)
PlayItem
SubPlayItem
〈INPUT〉
TIME
INFORMATION
IN_time(PTS)
STC_sequence_id
OUT_time(PTS)
STC_sequence_id
IN_time(PTS)
STC_sequence_id
OUT_time(PTS)
STC_sequence_id
〈OUTPUT〉
ADDRESS
INFORMATION
CPI(EP_map)
STC_sequence
CPI(EP_map)
STC_sequence
RELATIVE SOURCE
PACKET NUMBER
RELATIVE SOURCE
PACKET NUMBER
Clip AV STREAM FILE
Clip AV STREAM FILE

FIG.63

PlayList
(TU_map type)
PlayItem
PlayItem
〈INPUT〉
TIME
INFORMATION
IN_time
(Arrival Time)
OUT_time
(Arrival Time)
IN_time
(Arrival Time)
OUT_time
(Arrival Time)
CPI(TU_map)
CPI(TU_map)
〈OUTPUT〉
ADDRESS
INFORMATION
RELATIVE SOURCE
PACKET NUMBER
RELATIVE SOURCE
PACKET NUMBER
Clip AV STREAM FILE
Clip AV STREAM FILE

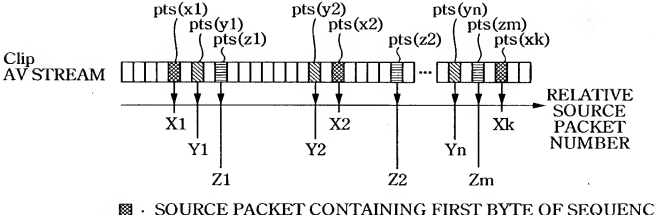
FIG.64

SYNTAX	NUMBER OF BYTES	ABBREVIATION
CPI(){		
version_number	8*4	bslbf
length	32	uimsbf
reserved	15	bslbf
CPI_type	1	bslbf
if (CPI_type==0)		
EP_map()		
else		
TU_map()		
}		

FIG.65

CPI_type	MEANING
0	EP map type
1	TU map type

FIG.66



EP_map

number_of_stream/PIDs=3

EP_map_for_ one_stream_PID(0)

stream_PID(0)=x
num_EP_entries(0)=k

PTS_EP start	RSPN_EP start
pts(x1)	X1
pts(x2)	X2
...	...
pts(xk)	Xk

EP_map_for_ one_stream_PID(1)

stream_PID(1)=y
num_EP_entries(1)=n

PTS_EP start	RSPN_EP start
pts(y1)	Y1
pts(y2)	Y2
...	...
pts(yn)	Yn

EP_map_for_ one_stream_PID(2)

stream_PID(2)=z
num_EP_entries(2)=m

PTS_EP start	RSPN_EP start
pts(z1)	Z1
pts(z2)	Z2
...	...
pts(zm)	Zm

FIG.67

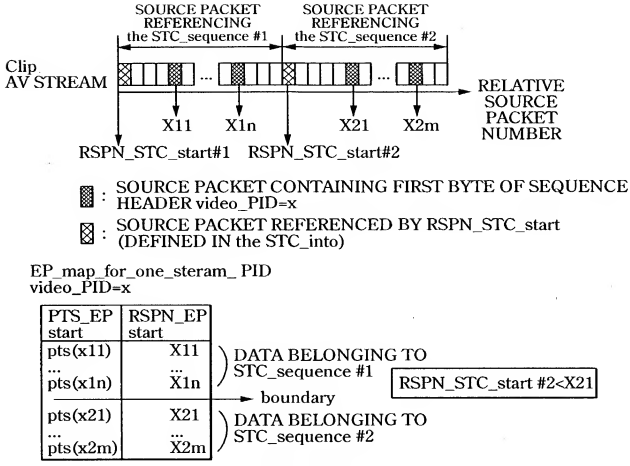

SOURCE PACKET REFERENCING the STC_sequence #1
SOURCE PACKET REFERENCING the STC_sequence #2
Clip AV STREAM
RELATIVE SOURCE PACKET NUMBER
X11
X1n
X21
X2m
RSPN_STC_start#1
RSPN_STC_start#2
: SOURCE PACKET CONTAINING FIRST BYTE OF SEQUENCE HEADER video_PID=x
: SOURCE PACKET REFERENCED BY RSPN_STC_start (DEFINED IN the STC_into)
EP_map_for_one_steram_ PID video_PID=x
PTS_EP start
RSPN_EP start
pts(x11)
pts(x1n)
X11
X1n
DATA BELONGING TO STC_sequence #1
RSPN_STC_start #2<X21
boundary
pts(x21)
pts(x2m)
X21
X2m
DATA BELONGING TO STC_sequence #2

FIG.68

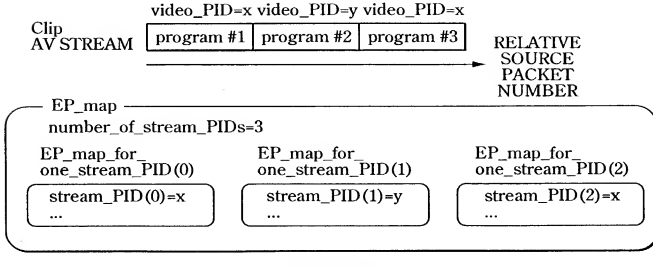

video_PID=x video_PID=y video_PID=x
Clip AV STREAM
program #1
program #2
program #3
RELATIVE SOURCE PACKET NUMBER
EP_map
number_of_stream_PIDs=3
EP_map_for_ one_stream_PID(0)
stream_PID(0)=x
EP_map_for_ one_stream_PID(1)
stream_PID(1)=y
EP_map_for_ one_stream_PID(2)
stream_PID(2)=x

FIG.69

SYNTAX	NUMBER OF BYTES	ABBREVIATION
EP_map(){		
reserved	12	bslbf
EP_type	4	uimsbf
number_of_stream_PIDs	16	uimsbf
for (k=0;k<*number_of_stream_PIDs*;k++){		
stream_PID(*k*)	16	bslbf
num_EP_entries(*k*)	32	uimsbf
EP_map_for_one_stream_PID_Start_address(*k*)	32	uimsbf
}		
for (i=0;i<X;i++){		
padding_word	16	bslbf
}		
for (k=0;k<*number_of_stream_PIDs*;k++){		
EP_map_for_one_stream_PID(*num_EP_entries(k)*)		
for (i=0;i<Y;i++){		
padding_word	16	bslbf
}		
}		
}		

FIG.70

EP_type	MEANING
0	video
1	audio
2-15	reserved

FIG.71

SYNTAX	NUMBER OF BYTES	ABBREVIATION
EP_map_for_one_stream_PID(*N*) {		
for (i=0;i<*N*;i++){		
PTS_EP_start	32	uimsbf
RSPN_EP_start	32	uimsbf
}		
}		

FIG.72

offset time
=TU_start_
time(0)
=TU_start_
time(1)
=TU_start_
time(N)
T_map
time axis
time_unit #0
time_unit #1
time_unit #N
SOURCE PACKET
ARRIVAL TIME CLOCK
INPUT
SOURCE
PACKET
Clip
AV STREAM
ADDRESS
(RELATIVE SOURCE PACKET NUMBER)
RSPN
time_unit
start #0
RSPN
time_unit
start #1
RSPN
time_unit
start #N

FIG.73

SYNTAX	NUMBER OF BYTES	ABBREVIATION
TU_map() {		
offset_time	32	bslbf
time_unit_size	32	uimsbf
number_of_time_unit_entries	32	uimsbf
for (k=0;k<*number_of_time_unit_entries*;k++)		
RSPN_time_unit_start	32	uimsbf
}		

FIG.74

SYNTAX	NUMBER OF BYTES	ABBREVIATION
ClipMark(){		
version_number	8*4	bslbf
length	32	uimsbf
number_of_Clip_marks	16	uimsbf
for (i=0; i<*number_of_clip_marks*; i++){		
reserved	8	bslbf
mark_type	8	bslbf
mark_time_stamp	32	uimsbf
STC_sequence_id	8	uimsbf
reserved	24	bslbf
character_set	8	bslbf
name_length	8	uimsbf
mark_name	8*256	bslbf
ref_thumbnail_index	16	uimsbf
}		
}		

FIG.75

Mark_type	MEANING	COMMENT
0x00-0x8F	reserved	Reserved for PlayListMark0
0x90	Event-start mark	MARK POINT INDICATING PROGRAM START POINT
0x91	Local event-start mark	MARK POINT INDICATING LOCAL SCENE IN PROGRAM
0x92	Scene-start mark	MARK POINT SHOWING SCENE CHANGE POINT
0x93-0xFF	reserved	

FIG.76

CPI_type in the PlayList()	SEMANTICS OF mark_time_stamp
EP_map type	mark_time_stamp MUST INDICATE UPPER 32 BITS OF 33 BIT LENGTH PTS CORRESPONDING TO PRESENTATION UNIT REFERENCED BY MARK.
TU_map type	mark_time_stamp MUST BE TIME ON TU_map_time_axis AND MUST BE ROUNDED TO time_unit PRECISION. mark_time_stamp IS CALCULATED BY FOLLOWING EQUATION: $mark\_time\_stamp = TU\_start\_time \%2^{32}$

FIG.77

SYNTAX	NUMBER OF BYTES	ABBREVIATION
ClipMark(){		
version_number	8*4	bslbf
length	32	uimsbf
number_of_Clip_marks	16	uimsbf
for (i=0; i<*number_of_Clip_marks*; i++){		
reserved	8	bslbf
mark_type	8	bslbf
reserved_for_MakerID	16	bslbf
mark_entry()		
representative_picture_entry()		
ref_thumbnail_index	16	uimsbf
}		
}		

FIG.78

Mark_type	MEANING	COMMENT
0x00-0x8F	reserved	Reserved for PlayListMark()
0x90	Event-start mark	MARK POINT INDICATING PROGRAM START POINT
0x91	Local event-start mark	MARK POINT INDICATING LOCAL SCENE IN PROGRAM
0x92	Scene-start mark	MARK POINT INDICATING SCENE START POINT
0x93	Scene-end mark	MARK POINT INDICATING SCENE END POINT
0x94	CM-start mark	MARK POINT INDICATING CM START POINT
0x95	CM-end mark	MARK POINT INDICATING CM END POINT
0x96-0xBF	DVR FORMAT IS RESERVED FOR FUTURE EXTENSION OF ClipMark	
0xC0-0xFF	ALLOCATBLE TO MARK USED IN MAKER-UNIQUE APPLICCATION	

FIG.79

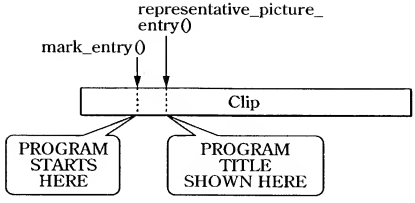


FIG.80

SYNTAX	NUMBER OF BYTES	ABBREVIATION
mark_entry()/representative_picture_entry(){		
mark_time_stamp	32	uimsbf
STC_sequence_id	8	uimsbf
reserved	24	bslbf
}		

FIG.81

SYNTAX	NUMBER OF BYTES	ABBREVIATION
mark_entry()/representative_picture_entry(){		
RSPN_ref_EP_start	32	uimsbf
offset_num_pictures	32	uimsbf
}		

FIG.82

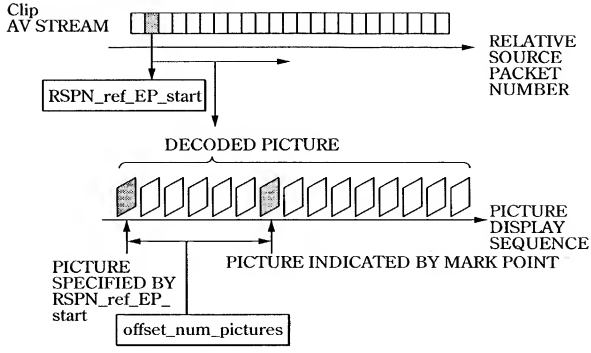


FIG.83

SYNTAX	NUMBER OF BYTES	ABBREVIATION
mark_entry()/representative_picture_entry(){		
RSPN_mark_point	32	uimsbf
}		

FIG.84

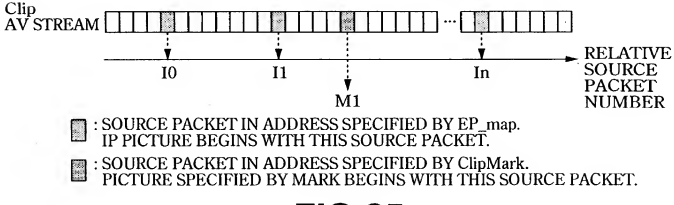


FIG.85

SYNTAX	NUMBER OF BYTES	ABBREVIATION
menu.thmb/mark.thmb() {		
reserved	256	bslbf
Thumbnail()		
for (i=0;i<N1;i++)		
padding_word	16	bslbf
}		

FIG.86

SYNTAX	NUMBER OF BYTES	ABBREVIATION
Thumbnail(){		
version_number	8*4	char
length	32	uimsbf
if (length !=0){		
tn_blocks_start_address	32	bslbf
number_of_thumbnails	16	uimsbf
tn_block_size	16	uimsbf
number_of_tn_blocks	16	uimsbf
reserved	16	bslbf
for (i=0; i<*number_of_thumbnails*; i++){		
thumbnail_index	16	uimsbf
thumbnail_picture_format	8	bslbf
reserved	8	bslbf
picture_data_size	32	uimsbf
start_tn_block_number	16	uimsbf
x_picture_length	16	uimsbf
y_picture_length	16	uimsbf
reserved	16	uimsbf
}		
stuffing_bytes	8*2*L1	bslbf
for(k=0; k<*number_of_tn_blocks*; k++){		
tn_block	tn_block_ size*1024*8	
}		
}		
}		

FIG.87

Thumbnail_picture_format	MEANING
0x00	MPEG-2 Video I-picture
0x01	DCF (restricted JPEG)
0x02	PNG
0x03-0xff	reserved

FIG.88

FIG.89A

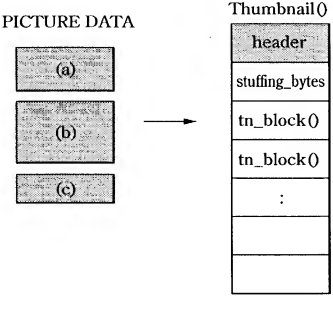
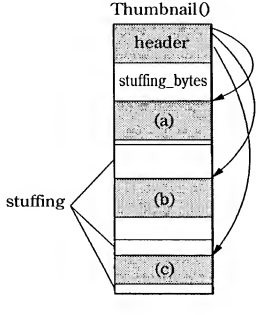


FIG.89B



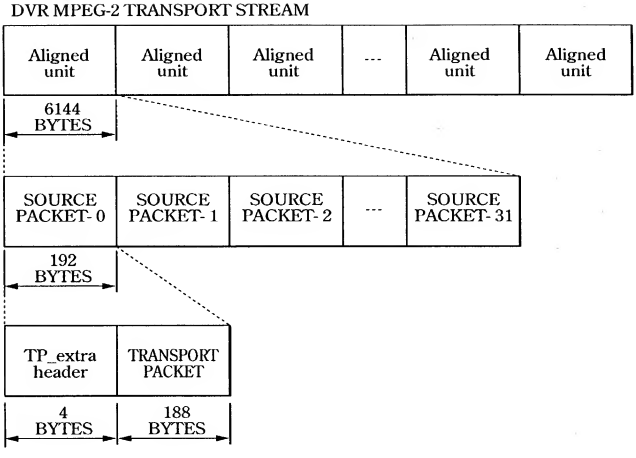


FIG.90

FIG.91

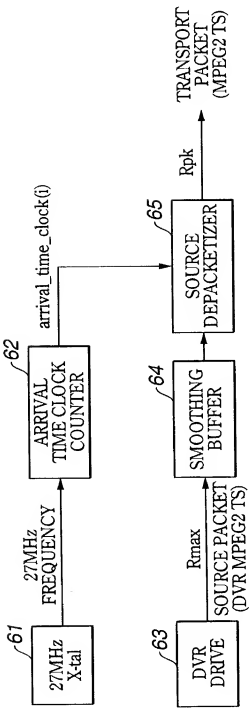

61
27MHz X-tal
27MHz FREQUENCY
62
ARRIVAL TIME CLOCK COUNTER
arrival_time_clock(i)
63
DVR DRIVE
Rmax
SOURCE PACKET (DVR MPEG2 TS)
64
SMOOTHING BUFFER
65
SOURCE DEPACKETIZER
Rpk
TRANSPORT PACKET (MPEG2 TS)

FIG.92

SYNTAX	NUMBER OF BYTES	ABBREVIATION
source_packet() {		
TP_extra_header()		
trasport_packet()		
}		

FIG.93

SYNTAX	NUMBER OF BYTES	ABBREVIATION
TP_extra_header() {		
copy_permission_indicator	2	uimsbf
arrival_time_stamp	30	uimsbf
}		

FIG.94

copy_permission _indicator	MEANING
00	copy free
01	no more copy
10	copy once
11	copy prohibited

FIG.95

FIG.96

IN_time 1
OUT_time 1
PREVIOUS
Playitem
Clip1
DECODE START
ADDRESS DETECTED
WITH CPI AND
IN_time 1
TS1
CONNECTION
POINT
IN_time 2
OUT_time 2
CURRENT
Playitem
Clip2
DECODE END
ADDRESS DETECTED
WITH CPI AND
OUT_time 2
TS2

FIG.97

Clip1
B0
B1
B2
B3
B4
B5
B6
B7
B8
B9
B10
B11
B12
B13
DISPLAY SEQUENCE
OUT_time1
REQUESTED CONNECTING POINT
Clip2
b0
b1
b2
b3
b4
b5
b6
b7
b8
b9
b10
b11
b12
b13
DISPLAY SEQUENCE
IN_time2

FIG.98

Clip1
i2
B0
B1
P5
B3
B4
P6
B6
B7
...
(DECODING SEQUENCE)
JUMP
RSPN_exit_from
_previous_Clip
DATA COPIED
FROM clip-1
RE-ENCODED
PICTURE
CONNECTION
POINT
PICTURE ENCODING
TYPE IN ORIGINAL Clip
DATA COPIED
FROM clip-2
RE-ENCODED
PICTURE
VIDEO OF
Bridge Sequence
BEFORE
SPN_arrival_time_
discontinuity
VIDEO OF Bridge Sequence
BEFORE SPN_arrival_
time_discontinuity
JUMP
RSPN_enter_to_
current_Clip
Clip2
...
i2
b0
b1
p5
b3
b4
p8
b6
b7
i11
b9
b10
p14
b12
b13

FIG.99

FIG.100

FIG.101

RSPN_exit_
from_previous
_Clip
RSPN_enter_
to_current_
Clip
Clip1
Clip2
Bridge-Clip
DATA ARRAYED
ON DISC
HALF
FRAGMENT
HALF
FRAGMENT
HALF
FRAGMENT
HALF
FRAGMENT
HALF
FRAGMENT

FIG.102

CONNECTION POINT
Clip2
Clip1
HALF FRAGMENT
HALF FRAGMENT
HALF FRAGMENT
HALF FRAGMENT
DATA ARRAYED ON DISC

FIG.103

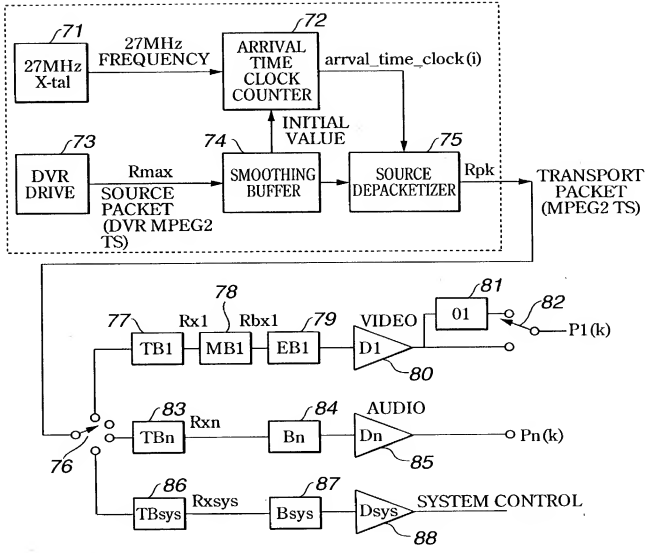

71
27MHz X-tal
27MHz FREQUENCY
72
ARRIVAL TIME CLOCK COUNTER
arrval_time_clock(i)
INITIAL VALUE
73
DVR DRIVE
Rmax
SOURCE PACKET (DVR MPEG2 TS)
74
SMOOTHING BUFFER
75
SOURCE DEPACKETIZER
Rpk
TRANSPORT PACKET (MPEG2 TS)
78
77
Rx1
Rbx1
79
VIDEO
81
01
82
P1(k)
TB1
MB1
EB1
D1
80
83
TBn
Rxn
84
Bn
AUDIO
Dn
Pn(k)
85
76
86
TBsys
Rxsys
87
Bsys
Dsys
SYSTEM CONTROL
88

FIG.104

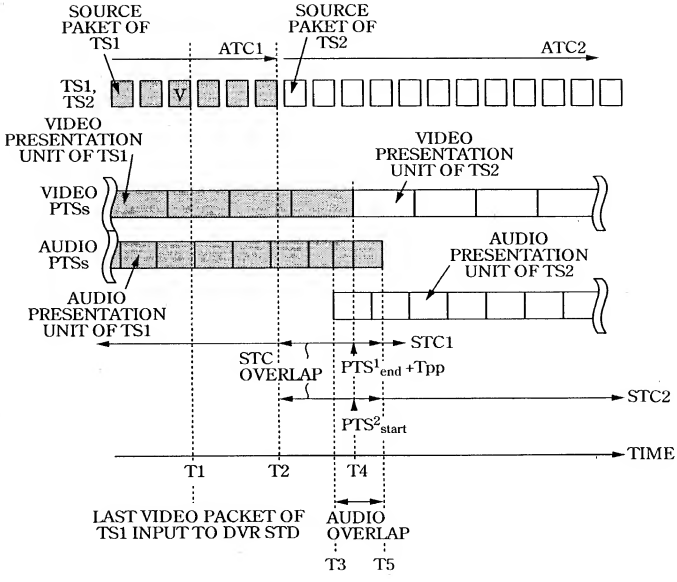

SOURCE PAKET OF TS1
ATC1
SOURCE PAKET OF TS2
ATC2
TS1, TS2
V
VIDEO PRESENTATION UNIT OF TS1
VIDEO PRESENTATION UNIT OF TS2
VIDEO PTSs
AUDIO PTSs
AUDIO PRESENTATION UNIT OF TS2
AUDIO PRESENTATION UNIT OF TS1
STC1
STC OVERLAP
PTS1end +Tpp
STC2
PTS2start
TIME
T1
T2
T4
LAST VIDEO PACKET OF TS1 INPUT TO DVR STD
AUDIO OVERLAP
T3
T5

FIG.105

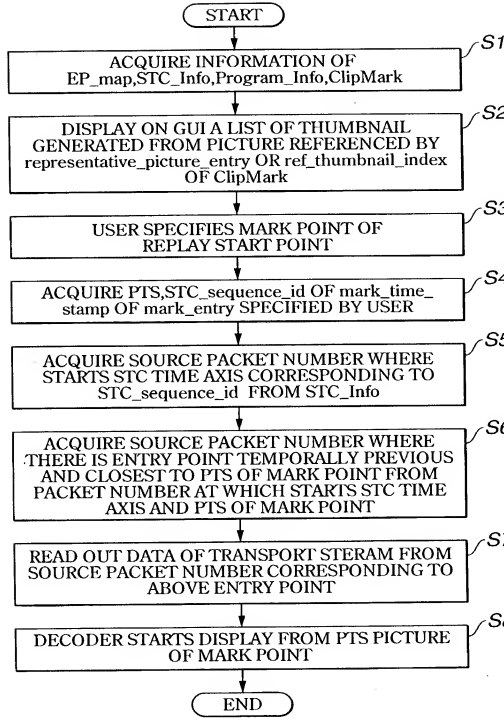

START
ACQUIRE INFORMATION OF EP_map,STC_Info,Program_Info,ClipMark
S1
DISPLAY ON GUI A LIST OF THUMBNAIL GENERATED FROM PICTURE REFERENCED BY representative_picture_entry OR ref_thumbnail_index OF ClipMark
S2
USER SPECIFIES MARK POINT OF REPLAY START POINT
S3
ACQUIRE PTS,STC_sequence_id OF mark_time_stamp OF mark_entry SPECIFIED BY USER
S4
ACQUIRE SOURCE PACKET NUMBER WHERE STARTS STC TIME AXIS CORRESPONDING TO STC_sequence_id FROM STC_Info
S5
ACQUIRE SOURCE PACKET NUMBER WHERE THERE IS ENTRY POINT TEMPORALLY PREVIOUS AND CLOSEST TO PTS OF MARK POINT FROM PACKET NUMBER AT WHICH STARTS STC TIME AXIS AND PTS OF MARK POINT
S6
READ OUT DATA OF TRANSPORT STERAM FROM SOURCE PACKET NUMBER CORRESPONDING TO ABOVE ENTRY POINT
S7
DECODER STARTS DISPLAY FROM PTS PICTURE OF MARK POINT
S8
END

FIG.106

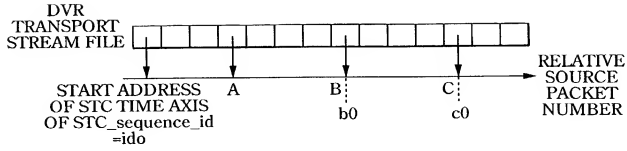


FIG.107

EP_map

RSPN_EP_ start	PTS_EP_ start
...	...
A	PTS(A)
B	PTS(B)
C	PTS(C)
...	...

FIG.108

ClipMark

Mark_type	mark_entry		representative _ picture _entry	
	Mark_ Time_stamp	STC_sequence_ id	Mark_ Time_stamp	STC_sequence_ id
...	...	...		
0x92(scene start)	PTS(a1)	id0	PTS(a2)	id0
0x94(CM start)	PTS(b0)	id0	PTS(b0)	id0
0x95(CM end)	PTS(c0)	id0	PTS(c0)	id0
...	...	...	...	...

FIG.109

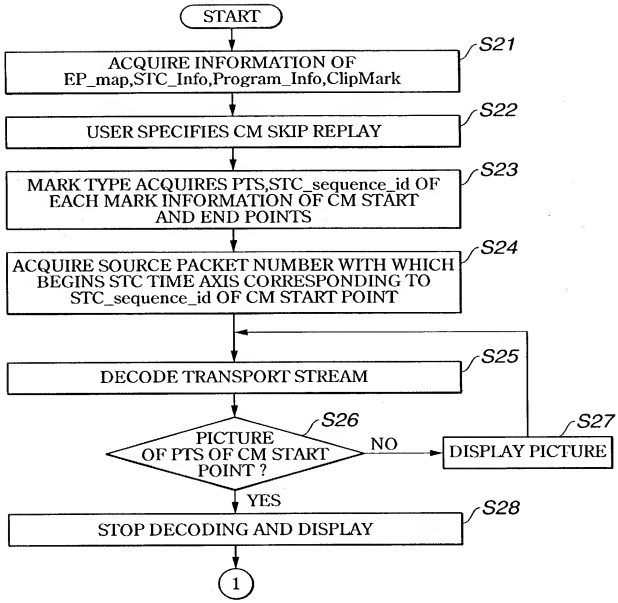

START
ACQUIRE INFORMATION OF EP_map,STC_Info,Program_Info,ClipMark
S21
USER SPECIFIES CM SKIP REPLAY
S22
MARK TYPE ACQUIRES PTS,STC_sequence_id OF EACH MARK INFORMATION OF CM START AND END POINTS
S23
ACQUIRE SOURCE PACKET NUMBER WITH WHICH BEGINS STC TIME AXIS CORRESPONDING TO STC_sequence_id OF CM START POINT
S24
DECODE TRANSPORT STREAM
S25
PICTURE OF PTS OF CM START POINT ?
S26
NO
DISPLAY PICTURE
S27
YES
STOP DECODING AND DISPLAY
S28
1

FIG.110

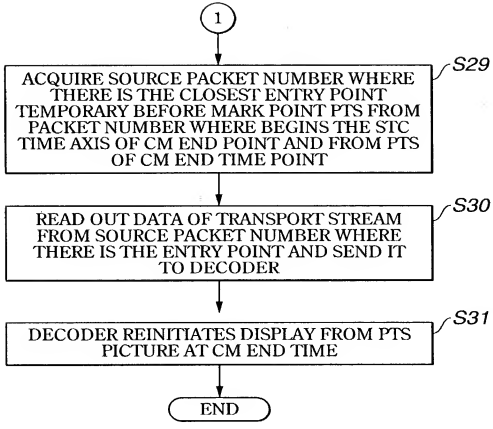

1
ACQUIRE SOURCE PACKET NUMBER WHERE THERE IS THE CLOSEST ENTRY POINT TEMPORARY BEFORE MARK POINT PTS FROM PACKET NUMBER WHERE BEGINS THE STC TIME AXIS OF CM END POINT AND FROM PTS OF CM END TIME POINT
S29
READ OUT DATA OF TRANSPORT STREAM FROM SOURCE PACKET NUMBER WHERE THERE IS THE ENTRY POINT AND SEND IT TO DECODER
S30
DECODER REINITIATES DISPLAY FROM PTS PICTURE AT CM END TIME
S31
END

FIG.111

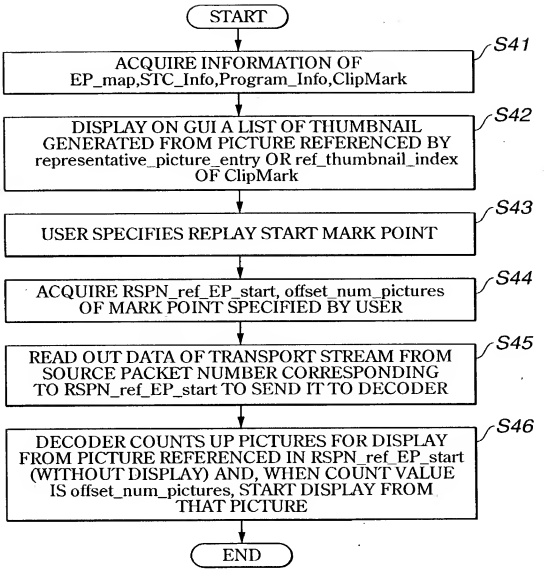

START
S41
ACQUIRE INFORMATION OF EP_map,STC_Info,Program_Info,ClipMark
S42
DISPLAY ON GUI A LIST OF THUMBNAIL GENERATED FROM PICTURE REFERENCED BY representative_picture_entry OR ref_thumbnail_index OF ClipMark
S43
USER SPECIFIES REPLAY START MARK POINT
S44
ACQUIRE RSPN_ref_EP_start, offset_num_pictures OF MARK POINT SPECIFIED BY USER
S45
READ OUT DATA OF TRANSPORT STREAM FROM SOURCE PACKET NUMBER CORRESPONDING TO RSPN_ref_EP_start TO SEND IT TO DECODER
S46
DECODER COUNTS UP PICTURES FOR DISPLAY FROM PICTURE REFERENCED IN RSPN_ref_EP_start (WITHOUT DISPLAY) AND, WHEN COUNT VALUE IS offset_num_pictures, START DISPLAY FROM THAT PICTURE
END

FIG.112

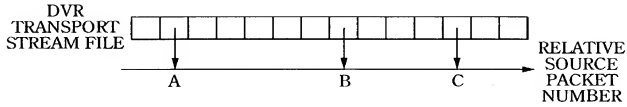


FIG.113

EP_map

RSPN_EP_ start	PTS_EP_ start
...	...
A	PTS(A)
B	PTS(B)
C	PTS(C)
...	...

FIG.114

ClipMark

mark_type	mark_entry		representative _ picture _entry	
	RSPN_ref_EP_ start	offset_num_ pictures	RSPN_ref_EP_ start	offset_num_ pictures
...	...	...	...	...
0x92(scene start)	A	M1	A	M2
0x94(CM start)	B	N1	B	N1
0x95(CM end)	C	N2	C	N2
...	...	...	...	...

FIG.115

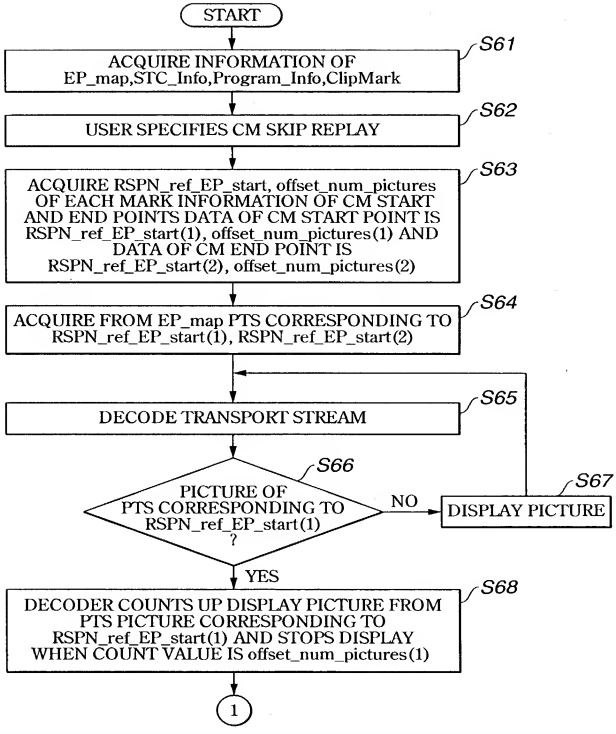

START
ACQUIRE INFORMATION OF EP_map,STC_Info,Program_Info,ClipMark
S61
USER SPECIFIES CM SKIP REPLAY
S62
ACQUIRE RSPN_ref_EP_start, offset_num_pictures OF EACH MARK INFORMATION OF CM START AND END POINTS DATA OF CM START POINT IS RSPN_ref_EP_start(1), offset_num_pictures(1) AND DATA OF CM END POINT IS RSPN_ref_EP_start(2), offset_num_pictures(2)
S63
ACQUIRE FROM EP_map PTS CORRESPONDING TO RSPN_ref_EP_start(1), RSPN_ref_EP_start(2)
S64
DECODE TRANSPORT STREAM
S65
PICTURE OF PTS CORRESPONDING TO RSPN_ref_EP_start(1) ?
S66
NO
DISPLAY PICTURE
S67
YES
DECODER COUNTS UP DISPLAY PICTURE FROM PTS PICTURE CORRESPONDING TO RSPN_ref_EP_start(1) AND STOPS DISPLAY WHEN COUNT VALUE IS offset_num_pictures(1)
S68
1

FIG.116

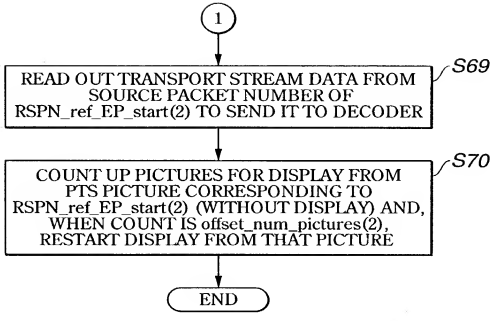

1
READ OUT TRANSPORT STREAM DATA FROM SOURCE PACKET NUMBER OF RSPN_ref_EP_start(2) TO SEND IT TO DECODER
S69
COUNT UP PICTURES FOR DISPLAY FROM PTS PICTURE CORRESPONDING TO RSPN_ref_EP_start(2) (WITHOUT DISPLAY) AND, WHEN COUNT IS offset_num_pictures(2), RESTART DISPLAY FROM THAT PICTURE
S70
END

FIG.117

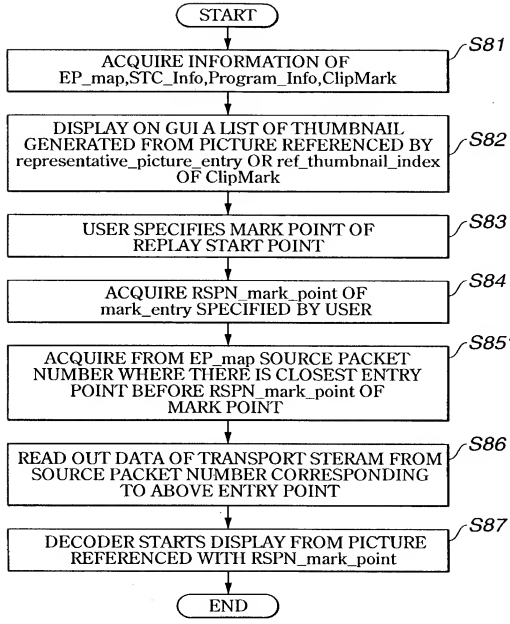

START
ACQUIRE INFORMATION OF EP_map,STC_Info,Program_Info,ClipMark
S81
DISPLAY ON GUI A LIST OF THUMBNAIL GENERATED FROM PICTURE REFERENCED BY representative_picture_entry OR ref_thumbnail_index OF ClipMark
S82
USER SPECIFIES MARK POINT OF REPLAY START POINT
S83
ACQUIRE RSPN_mark_point OF mark_entry SPECIFIED BY USER
S84
ACQUIRE FROM EP_map SOURCE PACKET NUMBER WHERE THERE IS CLOSEST ENTRY POINT BEFORE RSPN_mark_point OF MARK POINT
S85
READ OUT DATA OF TRANSPORT STERAM FROM SOURCE PACKET NUMBER CORRESPONDING TO ABOVE ENTRY POINT
S86
DECODER STARTS DISPLAY FROM PICTURE REFERENCED WITH RSPN_mark_point
S87
END

FIG.118

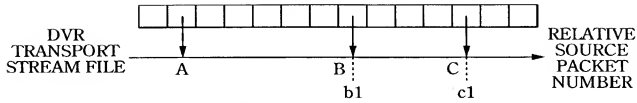


FIG.119

EP_map

RSPN_EP_ start	PTS_EP_ start
...	...
A	PTS(A)
B	PTS(B)
C	PTS(C)
...	...

FIG.120

ClipMark

mark_type	mark_entry	representative _ picture _entry
	RSPN_mark_ point	RSPN_mark_point
...	...	...
0x92(scene start)	a1	a2
0x94(CM start)	b1	b1
0x95(CM end)	c1	c1
...	...	...

FIG.121

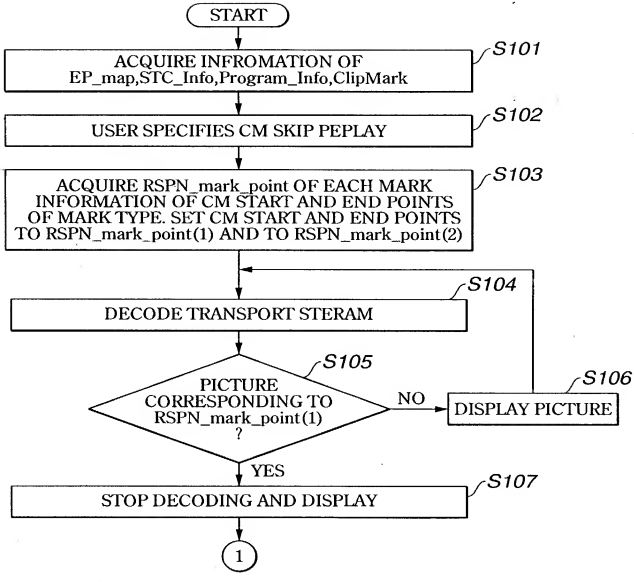

START
ACQUIRE INFROMATION OF EP_map,STC_Info,Program_Info,ClipMark
S101
USER SPECIFIES CM SKIP PEPLAY
S102
ACQUIRE RSPN_mark_point OF EACH MARK INFORMATION OF CM START AND END POINTS OF MARK TYPE. SET CM START AND END POINTS TO RSPN_mark_point(1) AND TO RSPN_mark_point(2)
S103
S104
DECODE TRANSPORT STERAM
S105
PICTURE CORRESPONDING TO RSPN_mark_point(1) ?
NO
S106
DISPLAY PICTURE
YES
S107
STOP DECODING AND DISPLAY
1

FIG.122

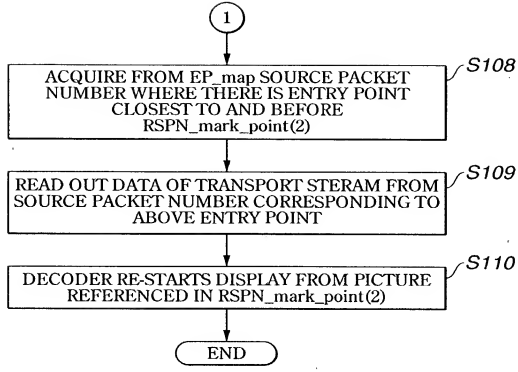

1
ACQUIRE FROM EP_map SOURCE PACKET NUMBER WHERE THERE IS ENTRY POINT CLOSEST TO AND BEFORE RSPN_mark_point(2)
S108
READ OUT DATA OF TRANSPORT STERAM FROM SOURCE PACKET NUMBER CORRESPONDING TO ABOVE ENTRY POINT
S109
DECODER RE-STARTS DISPLAY FROM PICTURE REFERENCED IN RSPN_mark_point(2)
S110
END

FIG.123

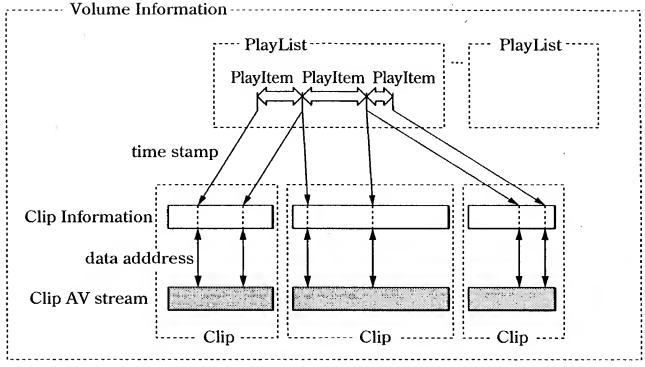


FIG.124

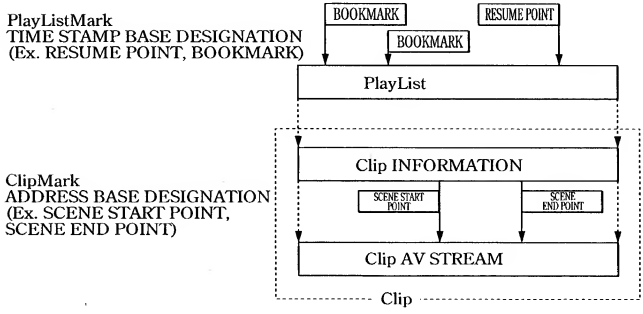


FIG.125

SYNTAX	NUMBER OF BYTES	ABBREVIATION
ClipMark(){		
version_number	8*4	bslbf
length	32	uimsbf
number_of_Clip_marks	16	uimsbf
for (i=0; i<*number_of_Clip_marks*; i++){		
reserved	8	bslbf
mark_type	8	bslbf
RSPN_mark	32	uimsbf
reserved	32	bslbf
ref_thumbnail_index	16	uimsbf
}		
}		

FIG.126

SYNTAX	NUMBER OF BYTES	ABBREVIATION
ClipMark(){		
version_number	8*4	bslbf
length	32	uimsbf
number_of_Clip_marks	16	uimsbf
for (i=0; i<*number_of_Clip_marks*; i++){		
reserved	8	bslbf
mark_type	8	bslbf
RSPN_ref_EP_start	32	uimsbf
offset_num_pictures	32	uimsbf
ref_thumbnail_index	16	uimsbf
}		
}		

FIG.127

SYNTAX	NUMBER OF BYTES	ABBREVIATION
ClipInfo(){		
length	32	uimsbf
reserved_for_word_align	8	bslbf
Clip_service_type	8	uimsbf
Clip_stream_type	8	uimsbf
reserved_for_word_align	6	bslbf
transcode_mode_flag	1	bslbf
time_controlled_flag	1	bslbf
TS_average_rate	32	uimsbf
TS_recoding_rate	32	uimsbf
reserved_for_DVRsystem_use	144	bslbf
TS_type_info_block()		
}		

FIG.128

SYNTAX	NUMBER OF BYTES	ABBREVIATION
ProgramInfo(){		
length	32	uimsbf
reserved_for_word_align	8	bslbf
num_of_program_sequences	8	uimsbf
for (i=0;i<*num_of_program_sequences*;i++){		
SPN_program_sequences_start	32	uimsbf
program_map_PID	16	bslbf
num_of_streams_in_ps	8	uimsbf
num_of_groups	8	uimsbf
for (*stream_index=0;* *stream_index<num_of_streams_in_ps;* *stream_index++*){		
stream_PID	16	uimsbf
StreamCodingInfo()		
}		
if (*num_of_groups*>1){		
for (i=0;i<*num_of_groups*;i++){		
num_of_streams_in_group	8	uimsbf
for (k=0;k<*num_of_streams_in_group*;k++){		
stream_index	8	uimsbf
}		
if (num_of_streams_in_group%2==0){		
reserved_for_word_align	8	bslbf
}		
}		
}		
}		
}		

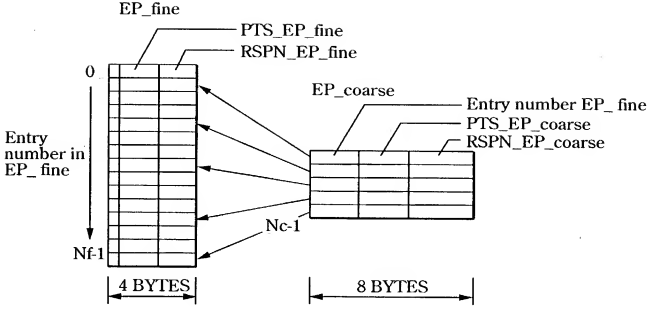
FIG.129

SYNTAX	NUMBER OF BYTES	ABBREVIATION
StreamCodingInfo(){		
length	8	bslbf
stream_coding_type	8	uimsbf
if (*stream_coding_type*==0x02){		
video_format	4	uimsbf
frame_rate	4	uimsbf
display_aspect_ratio	4	uimsbf
reserved_for_word_align	2	bslbf
cc_flag	1	uimsbf
original_video_format_flag	1	
if (*original_video_format_flag*==1){		
original_video_format	4	uimsbf
original_display_aspect_ratio	4	uimsbf
reserved_for_word_align	8	bslbf
}		
} else if(*stream_coding_type==0x03 //*		
stream_coding_type==0x04 //		
stream_coding_type==0x0F //		
stream_coding_type==0x80 //		
stream_coding_type==0x81 //		
audio_presentation_type	4	uimsbf
sampling_frequency	4	uimsbf
reserved_for_word_align	8	bslbf
}		
}		

FIG.130

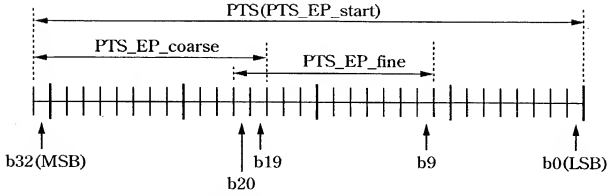
stream_coding_type	MEANING
0x00-0x01	FUTURE RESERVE
0x02	MPEG-1 OR MPEG-2 VIDEO STREAM
0x03	MPEG-1 AUDIO
0x04	MPEG-2 MULTI-CHANNEL AUDIO LOWER COMPATIBLE WITH MPEG-1
0x05	FUTURE RESERVE
0x06	TELETEXT DEFINED IN SESF OR DVB OR SUBTITLE DEFINED IN ISDB
0x07-0x09	FUTURE RESERVE
0x0A	ISO/IEC 13818-6 TYPE A
0x0B	ISO/IEC 13818-6 TYPE B
0x0C	ISO/IEC 13818-6 TYPE C
0x0D	ISO/IEC 13818-6 TYPE D
0x0E	FUTURE RESERVE
0x0F	MPEG-2AAC AUDIO HAVING ADTS TRANSPORT SYNTAX
0x10-0x7F	FUTURE RESERVE
0x08	SESF LPCM AUDIO
0x81	Dolby AC-3 AUDIO
0x82-0xFF	FUTURE RESERVE

FIG.131



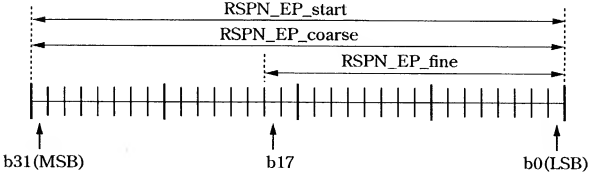
Nf IS ENTRY NUMBER IN EP-fine
Nc IS ENTRY NUMBER IN EP_coarse (Nc<Nf)

FIG.132



PTS :b0..b32(33-bit, 90kHz)
PTS_EP_fine :b9..b20(12-bit, Resolution=5.7msec and Wraparound in 23 seconds approximately)
PTS_EP_coarse :b19..b32(14-bit, Resolution=5.8sec and Wraparound in 26.5 hours approximately)

FIG.133



RSPN_EP_start :b0..b31(32-bit)
RSPN_EP_fine :b0..b17(18-bit, Wrap around in 50 Mbyte approximately in the AV stream file)
RSPN_EP_coarse :b0..b31(32-bit)

FIG.134

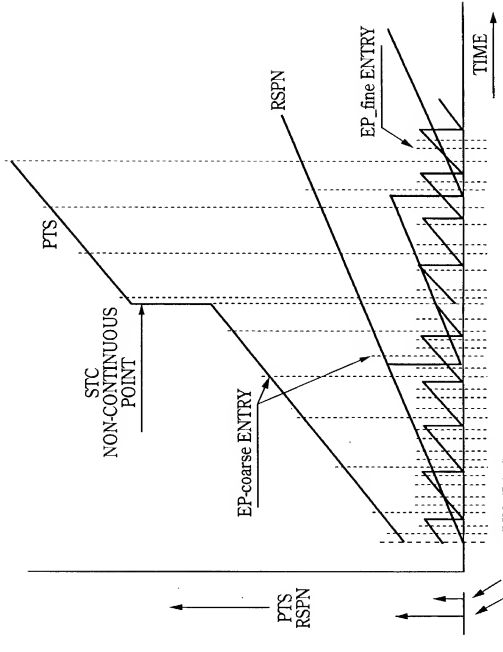

PTS
STC
NON-CONTINUOUS
POINT
EP-coarse ENTRY
RSPN
EP_fine ENTRY
TIME
PTS
RSPN
range PTS_EP_fine
range RSPN_EP_fine

FIG.135

SYNTAX	NUMBER OF BYTES	ABBREVIATION
EP_map(){		
reserved_for_word_align	8	bslbf
number_of_stream_PID_entries	8	uimsbf
for (k=0;k<*number_of_stream_PID_entries*;k++){		
stream_PID[*k*]	16	bslbf
reserved_for_word_align	10	bslbf
EP_stream_type[*k*]	4	uimsbf
num_EP_coarse_entries[*k*]	16	uimsbf
num_EP_fine_entries[*k*]	18	uimsbf
EP_map_for_one_stream_PID_start_address[*k*]	32	uimsbf
}		
for (i=0;i<X;i++){		
padding_word	16	bslbf
}		
for (k=0;k<*number_of_stream_PID_entries*;k++){		
EP_map_for_one_stream_PID (*EP_stream_type[k]; num_EP_coarse_entries[k]; num_EP_fine_entries[k]*)		
for (i=0;i<Y[k];i++){		
padding_word	16	bslbf
}		
}		
}		

FIG.136

EP_stream_type	MEANING
0	video type1
1	video type2
2	audio
3-15	reserved for future use

FIG.137

SYNTAX	NUMBER OF BYTES	ABBREVIATION
EP_map_for_one_stream_PID (*EP_stream_type,Nc,Nf*){		
EP_fine_table_start_address	32	uimsbf
for (i=0;i<*Nc*;i++){		
ref_to_EP_fine_id*[i]*	18	uimsbf
PTS_EP_coarse*[i]*	14	uimsbf
RSPN_EP_coarse*[i]*	32	uimsbf
}		
for (i=0;i<X;i++){		
padding_word	16	bslbf
}		
for (*EP_fine_id=0;* *EP_fine_id<Nf;* *EP_fine_id++*){		
EP_video_type*[EP_fine_id]*	2	
PTS_EP_fine*[EP_fine_id]*	12	uimsbf
RSPN_EP_fine*[EP_fine_id]*	18	uimsbf
}		
}		

FIG.138

	MEANING
0	VIDEO ACCESS UNIT AT ENTRY POINT IS I-PICTURE BEGINNING FROM SEQUENCE HEADER, THIS I PICTURE MAY BE PRECEDED BY GOP HEADER. SPN_EP_start INDICATES ADDRESS OF SOURCE PACKET CONTAINING BYTE 1 OF SEQUENCE HEADER CODE OF ACCESS UNIT.
1	VIDEO ACCESS UNIT AT ENTRY POINT IS P-PICTURE BEGINNING FROM SEQUENCE HEADER. SPN_EP_start INDICATES ADDRESS OF SOURCE PACKET CONTAINING BYTE 1 OF SEQUENCE HEADER CODE OF ACCESS UNIT.
2	VIDEO ACCESS UNIT AT ENTRY POINT IS I-PICTURE NOT BEGINNING FROM SEQUENCE HEADER, THIS I PICTURE MAY BE PRECEDED BY GOP HEADER. IF I PICTURE IS PRECEDED BY GOP HEADER, SPN_EP_start INDECATES ADDRESS OF SOURCE PACKET CONTAINING BYTE 1 OF GROUP START CODE OF ACCESS UNIT. IF I PICTURE IS NOT PRECEDED BY GOP HEADER, SPN_EP_start INDECATES ADDRESS OF SOURCE PACKET CONTAINING BYTE 1 OF GROUP START CODE OF ACCESS UNIT.
3	reserved for future use

FIG.139

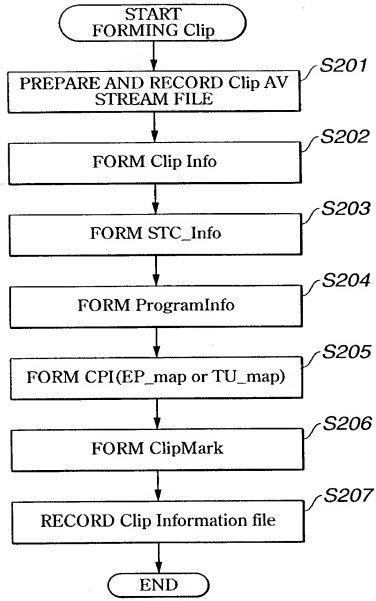

START
FORMING Clip
PREPARE AND RECORD Clip AV STREAM FILE
S201
FORM Clip Info
S202
FORM STC_Info
S203
FORM ProgramInfo
S204
FORM CPI(EP_map or TU_map)
S205
FORM ClipMark
S206
RECORD Clip Information file
S207
END

FIG.140

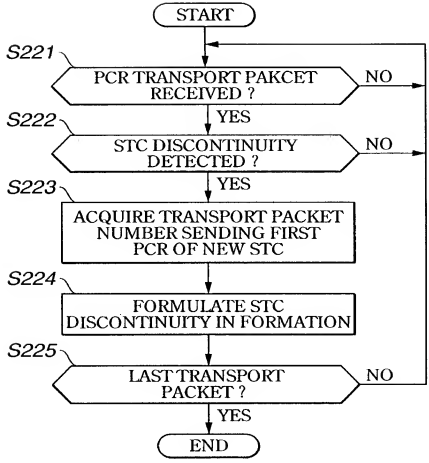

START
S221
PCR TRANSPORT PAKCET RECEIVED ?
NO
YES
S222
STC DISCONTINUITY DETECTED ?
NO
YES
S223
ACQUIRE TRANSPORT PACKET NUMBER SENDING FIRST PCR OF NEW STC
S224
FORMULATE STC DISCONTINUITY IN FORMATION
S225
LAST TRANSPORT PACKET ?
NO
YES
END

FIG.141

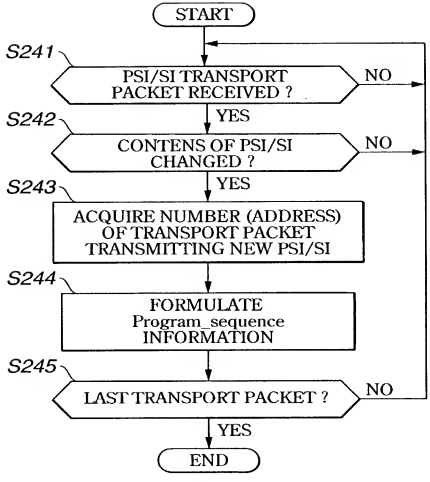

START
S241
PSI/SI TRANSPORT PACKET RECEIVED ?
NO
YES
S242
CONTENS OF PSI/SI CHANGED ?
NO
YES
S243
ACQUIRE NUMBER (ADDRESS) OF TRANSPORT PACKET TRANSMITTING NEW PSI/SI
S244
FORMULATE Program_sequence INFORMATION
S245
LAST TRANSPORT PACKET ?
NO
YES
END

FIG.142

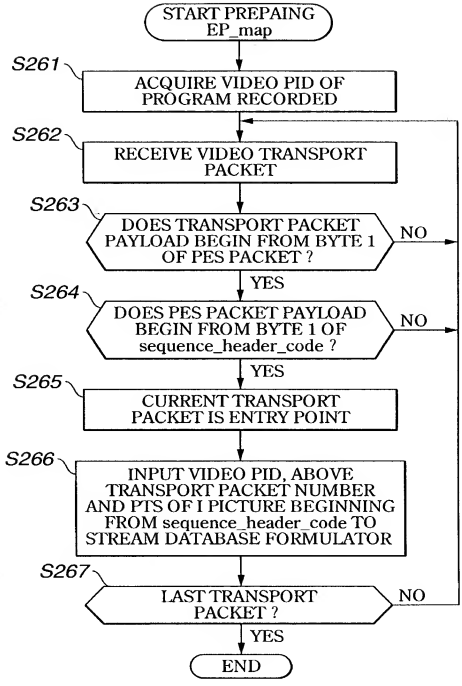

START PREPAING EP_map
S261
ACQUIRE VIDEO PID OF PROGRAM RECORDED
S262
RECEIVE VIDEO TRANSPORT PACKET
S263
DOES TRANSPORT PACKET PAYLOAD BEGIN FROM BYTE 1 OF PES PACKET ?
NO
YES
S264
DOES PES PACKET PAYLOAD BEGIN FROM BYTE 1 OF sequence_header_code ?
NO
YES
S265
CURRENT TRANSPORT PACKET IS ENTRY POINT
S266
INPUT VIDEO PID, ABOVE TRANSPORT PACKET NUMBER AND PTS OF I PICTURE BEGINNING FROM sequence_header_code TO STREAM DATABASE FORMULATOR
S267
LAST TRANSPORT PACKET ?
NO
YES
END

FIG.143

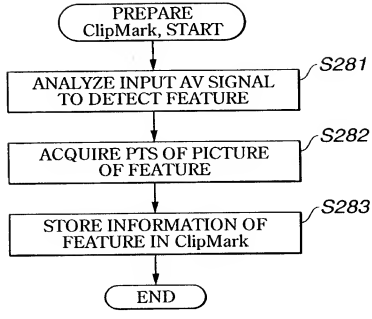

PREPARE
ClipMark, START
S281
ANALYZE INPUT AV SIGNAL
TO DETECT FEATURE
S282
ACQUIRE PTS OF PICTURE
OF FEATURE
S283
STORE INFORMATION OF
FEATURE IN ClipMark
END

FIG.144

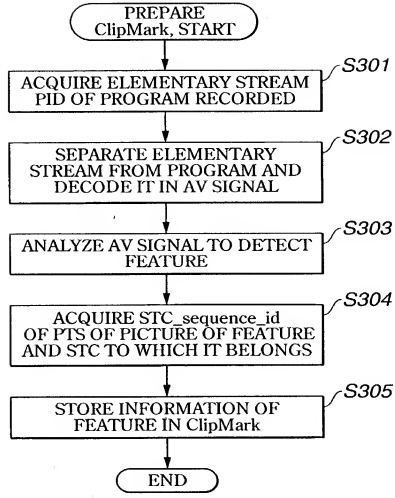

PREPARE ClipMark, START
ACQUIRE ELEMENTARY STREAM PID OF PROGRAM RECORDED
S301
SEPARATE ELEMENTARY STREAM FROM PROGRAM AND DECODE IT IN AV SIGNAL
S302
ANALYZE AV SIGNAL TO DETECT FEATURE
S303
ACQUIRE STC_sequence_id OF PTS OF PICTURE OF FEATURE AND STC TO WHICH IT BELONGS
S304
STORE INFORMATION OF FEATURE IN ClipMark
S305
END

FIG.145

EP_map
video_PID=X
PTS_EP_start RSPN_EP_start
...
video_PID=Y
PTS_EP_start RSPN_EP_start
...
Clip AV STREAM FILE
video_PID=X
video_PID=Y
I-pic
I-pic
I-pic
I-pic
I-pic
I-pic
I-pic
I PICURE SEARCH
(TRICK PLAY, CHAPTER SEARCH ETC)

FIG.146

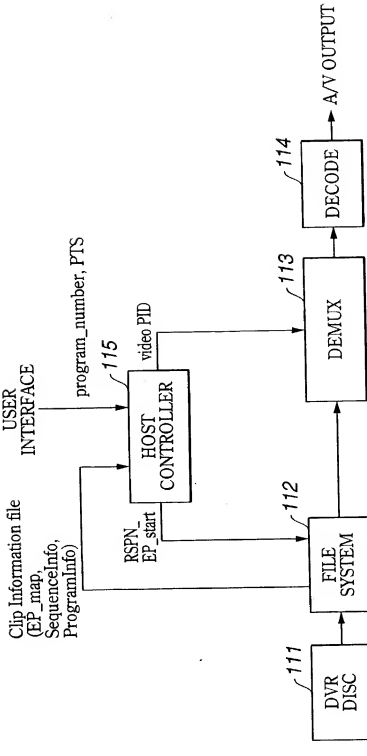


FIG.147

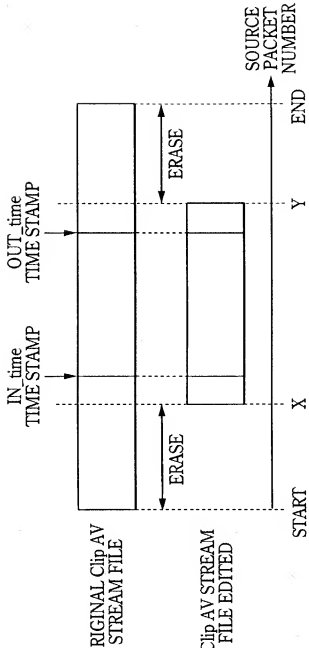

ORIGINAL Clip AV STREAM FILE
Clip AV STREAM FILE EDITED
IN_time TIME STAMP
OUT_time TIME STAMP
ERASE
ERASE
START
X
Y
END
SOURCE PACKET NUMBER

FIG.148

EP_map
PTS
EP_start
...
pts1
pts2
...
SPN
EP_start
...
ISA1
ISA2
...
pts1<pts2<IN_time PTS
IN_time PTS
>=100msec
pts1
pts2
SQH
PCR
PMT
PAT
SQH
ORIGINAL Clip AV STREAM FILE
Clip AV STREAM FILE EDITED
ERASE
ISA1
ISA2
SOURCE PACKET NUMBER
X POINT IS BOUNDARY OF ALIGNED UNIT

FIG.149

EP_map
PTS SPN
EP_start EP_start
... ...
pts4 ISA4
pts5... ISA5
...
I2 B0 B1 P5 B3 B4...
OUT_time PTS
pts4
pts5
SQH
SQH
SQH
SQH
ERASE
ORIGINAL Clip AV STREAM FILE
Clip AV STREAM FILE EDITED
ISA4
ISA5
SOURCE PACKET NUMBER
X POINT IS BOUNDARY OF ALIGNED UNIT

FIG.150

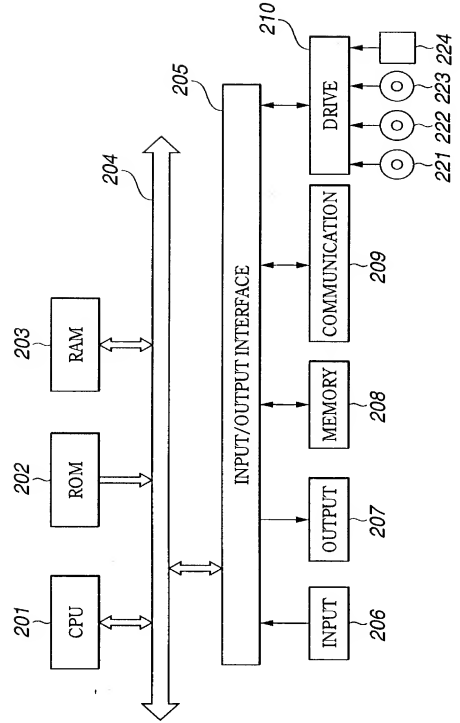

201
CPU
202
ROM
203
RAM
204
205
INPUT/OUTPUT INTERFACE
206
INPUT
207
OUTPUT
208
MEMORY
209
COMMUNICATION
210
DRIVE
221
222
223
224

FIG.151